Naval Research Laboratory

Washington, DC 20375-5320

NRL/FR/5740--95-9801

An Application of Clustering and Speed Discrimination to Tracking

JAMES F. SMITH
ANDY HUYNH
MOON W. KIM

Surface Electronic Warfare Systems Branch
Tactical Electronic Warfare Division

December 29, 1995

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave Blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 29, 1995	

4. TITLE AND SUBTITLE

An Application of Clustering and Speed Discrimination to Tracking

5. FUNDING NUMBERS

N001405WX40007
AA1751319.W236
57-6140-C6

6. AUTHOR(S)

James F. Smith, Andy Huynh, and Moon W. Kim

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/FR/5740--95-9801

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
Arlington, VA 22217-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

A tracking algorithm that combines clustering and speed discrimination is examined. A performance matrix is defined, and for simulated data, the algorithm is shown to have a success rate as high as 92%, in terms of its ability to extract emitter tracks from data. Conditions under which the algorithm succeeds or fails are analyzed. The algorithm is applied to real inorganic sensor data for ships and aircraft, and the results are discussed.

14. SUBJECT TERMS

Tracking
Clustering
Cluster Radius
Inorganic Sensors
Speed Discrimination
Radar Performance Matrix
Fuzzy Sets

15. NUMBER OF PAGES

29

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std 239-18
298-102

CONTENTS

AN APPLICATION OF CLUSTERING AND SPEED DISCRIMINATION TO TRACKING

1. INTRODUCTION

The problem considered in this report is to extract emitter tracks from experimentally obtained inorganic sensor data. The measurement system reports data about emissions of radar units aboard both ships and aircraft. This report describes a first attempt at the development of a heuristic algorithm that combines Euclidean clustering (EC), speed discrimination, and track formation for the processing of inorganic sensor data. Results are presented for both simulated and measured data. The full three component tracking algorithm will be called the Euclidean Clustering Speed Discrimination Track Formation (ECSDTF) algorithm. More sophisticated approaches to tracking [1-6] are currently under investigation.

The data consist of vectors, the elements of which represent measurements of various quantities: an indication of the radar type aboard the ship or aircraft being tracked (ID), the emitter's latitude, longitude, pulse width (PW), radio frequency (RF), pulse repetition interval (PRI), the type of PRI (PRI-type), time of intercept of the measured information, ellipse semi-major axis, and ellipse semi-minor axis. These data provide the input for the tracking algorithm.

Figure 1 is a histogram of ship data with fixed ID, PW, and PRI-type as observed by an inorganic sensor system. There were 202 observations over 25 hours. Most of the observations (90%) occurred over a 13.3 hour time window. The minimum separation between nonsimultaneously reported observations is 1 min, the maximum is 76 min, and the average is 4 min. The maximum number of observations in any minute within this window is 6.

Figure 2 is a histogram of observations made by the inorganic sensor system for an aircraft. There were 26 observations with the same ID, PW, and PRI-type over a little more than 12 hours. Most of the observations (about 90%) occurred over 7 hours. The minimum separation between nonsimultaneously reported observations is 1 min, the maximum is 111 min, and the average is 19.48 min. The maximum number of observations in any minute within this window is 2. The highest concentration of the data is in a 4 hour subwindow that contains more than 70% of the observations.

The relatively low observational density per unit time found in Figs. 1 and 2 seems to imply that data should be accumulated for seconds or minutes, at least, before analysis is carried out. The algorithm developed in this report is discussed in terms of its batch-mode characteristics and a simple recursive extension. In batch mode, data can be accumulated about an enemy ship or aircraft and, subsequently, the algorithm forms tracks. In the final stage, the tracks are updated in real-time using the recursive extension of the algorithm as new data arrive.

Manuscript approved August 28, 1995.

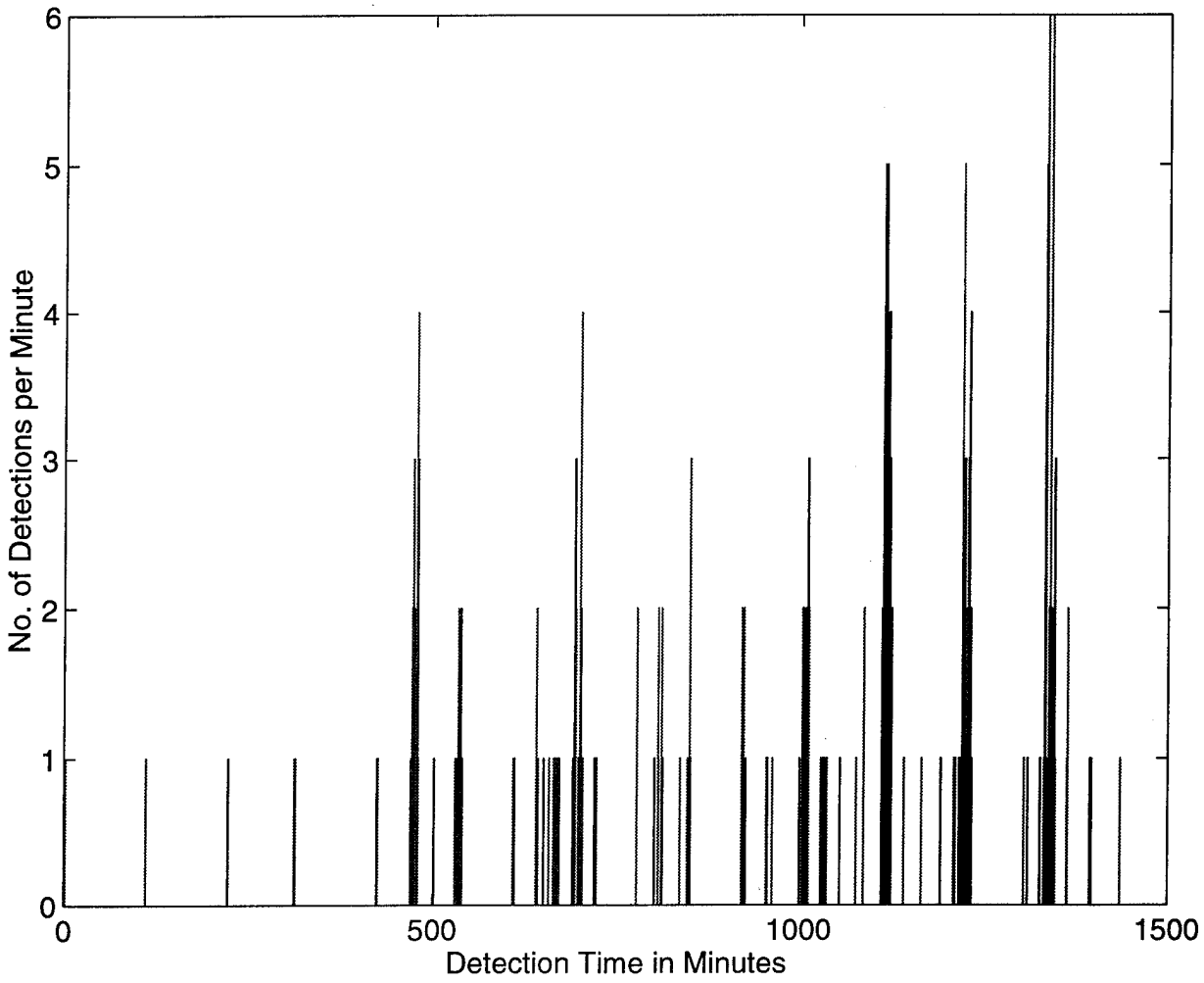

Fig. 1 — The number of ships of fixed ID, PW, and PRI-type detected by the inorganic sensor system in a 25 hour time window

The first component of the tracking algorithm is clustering, which is an operation that allows data to be grouped into classes defined by a similarity measure. By definition [7-8], given K objects, the algorithm forms N clusters such that with respect to the similarity measure, the members of each cluster have a greater similarity to each other than to the members of any other cluster.

The clustering algorithm used here is a very simple heuristic algorithm based on the Euclidean metric. More sophisticated algorithms based on neural nets [7], cost functions [8], or fuzzy sets[9] exist, but a simple approach is adopted for this first effort. Section 5 discusses more effective clustering procedures.

The second component of the tracking algorithm is speed discrimination. Speed discrimination restricts the set of potential paths to those that require the ships and aircraft being tracked to move at physically realistic speeds.

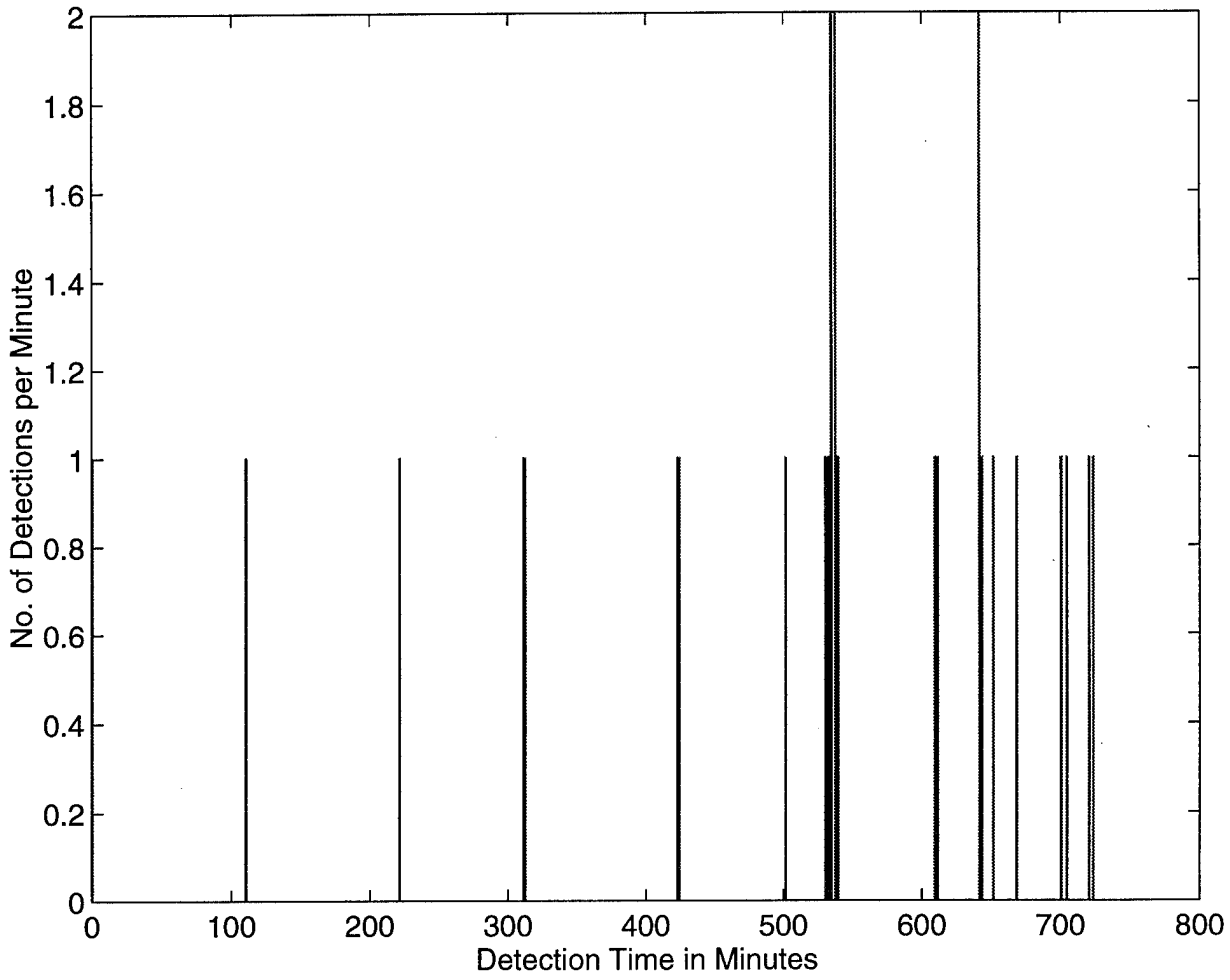

Fig. 2 — The number of aircraft of fixed ID, PW, and PRI-type detected per minute by the inorganic sensor system over about 12 hours

The third component, track formation, refers to the actual construction, i.e., plotting of all potential tracks. If the number of data points within a cluster is large, generally there will be many potential tracks, and it becomes convenient for visualization purposes to plot only one track. Also, when tracks are being extracted so they can be passed to another algorithm as input, it may be necessary to extract the one best track per cluster or subcluster. One convenient track to deal with is the time-nearest-neighbor-track, which is defined below.

Section 2 describes and analyzes the clustering, speed discrimination, and track formation algorithms. The section also defines a performance matrix and introduces the concept of the time-nearest-neighbor track. Section 3 discusses the application of the algorithm to simulated ship data and measured inorganic ship and aircraft data. The performance matrix is used to examine the effectiveness of the tracking algorithm for a simulated case. Section 4 provides conclusions and Section 5 discusses future extensions of the tracking algorithm.

2. DESCRIPTION AND EXAMINATION OF THE EUCLIDEAN CLUSTERING, SPEED DISCRIMINATION, AND TRACK FORMATION ALGORITHMS

The ECSDTF algorithm contains three major components: Euclidean clustering, speed discrimination, and track formation. The clustering algorithm is described in detail, followed by the speed discrimination and the track formation algorithms. A formula for estimating the cluster radius is developed. Also, a performance matrix is defined, which is useful in quantifying how successfully the algorithms extract emitter tracks from data. Finally, the concept of the time-nearest-neighbor-track is introduced.

Figure 3 gives an overview of the ECSDTF algorithm. As a first step, ID, PRI-type, and PW are fixed. Next, the algorithm produces clusters based on RF and PRI, and then it conducts speed discrimination within each cluster. Finally, the algorithm plots all the potential tracks within a cluster. In cases where many potential tracks exist, only the time-nearest-neighbor-track is plotted.

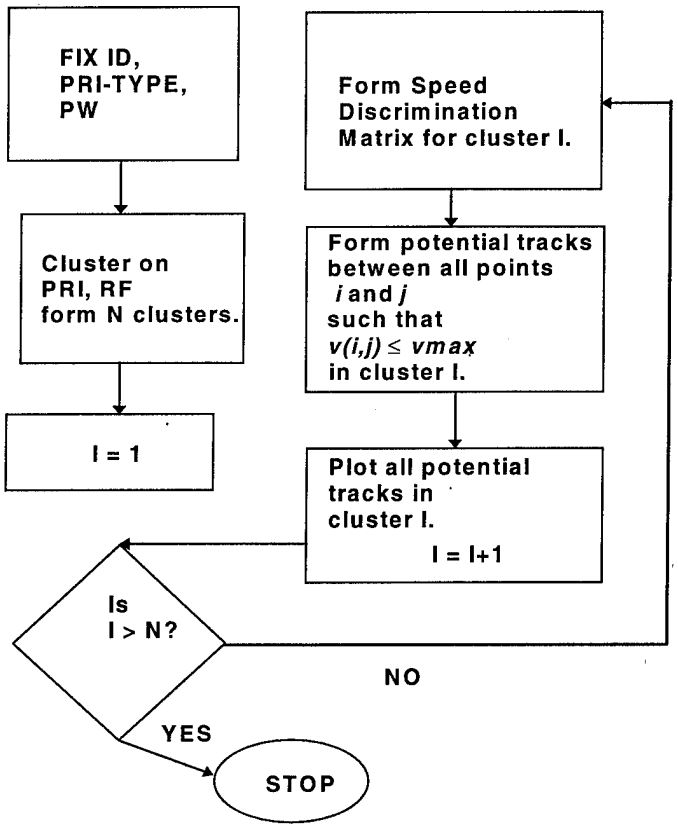

Fig. 3 — Flow chart of the tracking algorithm

2.1 The Euclidean Clustering Algorithm

Figure 4 is a flow chart of the Euclidean clustering algorithm. As a first step in understanding the algorithm, input and output requirements must be established. The input consists of the data to be clustered:

N_{min} = the minimum number of potential clusters;
N_{max} = the maximum number of potential clusters;
r_o = the initial cluster radius;

and

Δr = the incremental amount the radius is increased each time the algorithm uses N_{max}, clusters without fully covering the data.

When selecting the above input parameters, some trial and error may be required. If, for example, clustering is to be conducted on RF and PRI, the final cluster radius is generally within a factor of three of the root-mean-square (RMS) error, $\Delta(RF, PRI)$, which is defined as follows:

$$\Delta(RF, PRI) \equiv \left\{ \left[\frac{\Delta(RF)}{s(RF)} \right]^2 + \left[\frac{\Delta(PRI)}{s(PRI)} \right]^2 \right\}^{1/2},$$

where

$\Delta(RF) \equiv$ resolution limit of RF for detector,
$\Delta(PRI) \equiv$ resolution limit of PRI for detector,
$s(RF) \equiv$ RF scaling parameter,

and

$s(PRI) \equiv$ PRI scaling parameter.

When RF and PRI are independent Gaussian random variables, the RMS error is the RMS cluster radius. Thus, one would expect most of the data to fall within three standard deviations from the mean, as is observed in the simulated and measured cases considered in Section 3. The scaling parameters $s(RF)$ and $s(PRI)$ are used to place RF and PRI on the same scale. For the simulated and experimental cases of Section 3, $s(RF)$ and $s(PRI)$ are the standard deviations of the RF and PRI data values, respectively.

In practice, a value equal to one-tenth of $\Delta(RF, PRI)$ has been found effective for r_o and Δr. Selecting small values of r_o and Δr relative to $\Delta(RF, PRI)$ will increase the algorithm's run time, but is in general a safer procedure, since the clustering algorithm is designed to select the smallest cluster radius and the smallest number of clusters that cover the data. If values of r_o and Δr are selected that are much larger than $\Delta(RF, PRI)$, then the number of clusters that the algorithm determines to be correct may be considerably less than the number of objects being tracked. If values of r_o and Δr are selected that are too small and N_{max} is initially sufficiently large, then each data vector will fall into its own cluster.

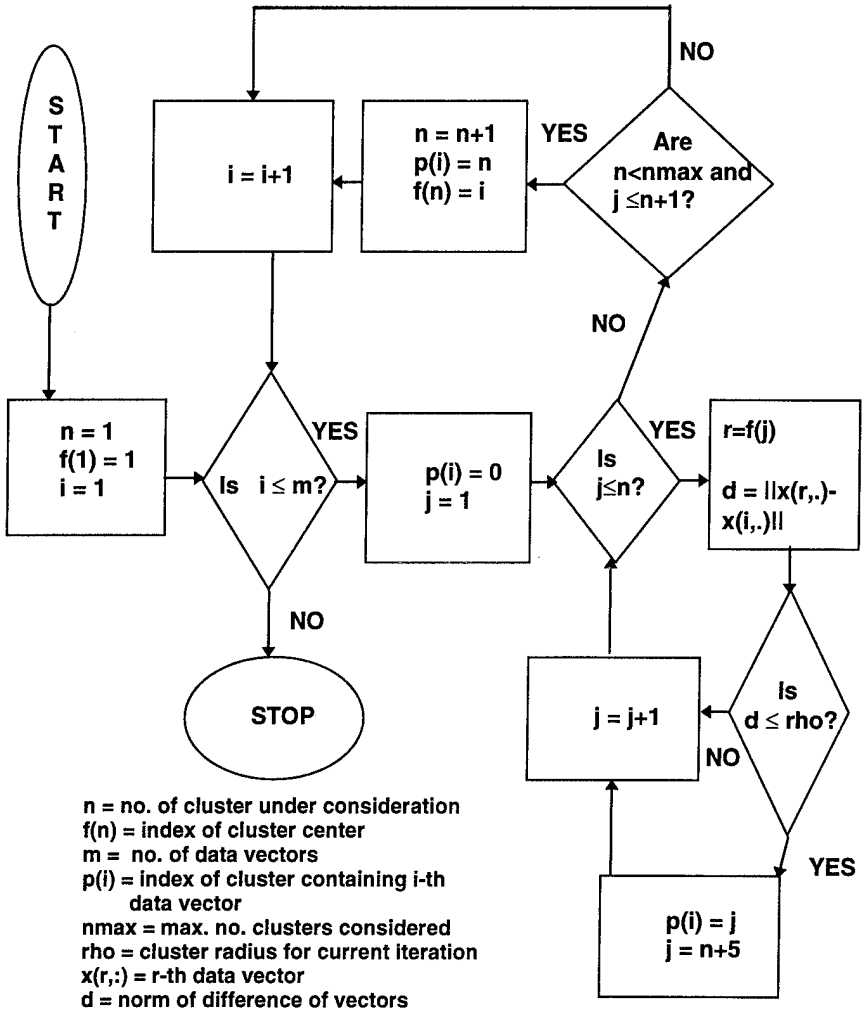

Fig. 4 — Flow chart of the clustering algorithm

The selection of N_{min} and N_{max} generally also requires some trial and error. If the clustering algorithm alone were successful in extracting tracks from data, simulated or experimental, there would be one and only one track per cluster, in which case,

$$N_{min} = N_{max} = N_{tracks} ,$$

where N_{tracks} is the true number of objects being tracked.

This can be understood by again considering RF and PRI to be independent Gaussian random processes. In this case, an ideal clustering algorithm capable of establishing cluster centers at (mean(RF), mean(PRI)) for each track; and also capable of suppressing outliers, could cover most of the data with N_{tracks} clusters of radii equal to $3 \times \Delta(RF, PRI)$.

Rarely is clustering successful in uniquely determining the number of tracks. As such, in practice, N_{min} and N_{max} are selected so that

$$N_{min} << N_{tracks} << N_{max}.$$

The final output of this process is clustered data. The algorithm is robust under selection of different values of N_{min} and N_{max} with the number of clusters varying by at most one. Also, the assignment of data points to clusters is not changed for most points. For lower dimensional cases (one to three dimensions), results can be displayed graphically. Examples of clustered data are given in Figs. 7, 13, and 17, for data corresponding to a simulated ship, a real ship, and a real aircraft, respectively.

Having described the input and desired output, the actual operation of the program can be examined. As an initial step in the clustering operation, the first data vector in the input file becomes the center of a ball of radius equal to the initial cluster radius. This ball is the first cluster. If the next data vector lies within this ball, it is classified as part of the same cluster, otherwise the algorithm forms another closed ball, centered on the second data vector. The same radius is maintained during this process. Each time the algorithm is about to form a new cluster, it makes sure that the maximum number of clusters has not been exceeded, in which case, it creates another cluster, with the previous unclassified data vector as its center. If the maximum number of clusters has been reached, the algorithm starts over with the minimum number of clusters, first increasing the cluster radius by Δr. The algorithm continues in this fashion, until all points have been classified.

In Fig. 5, the Euclidean clustering algorithm is applied to a simple example to illustrate the procedure. In the left-hand column the maximum number of clusters that are involved in that stage of the operation is given. The right-hand column has a sequence of diagrams showing the evolution of a simulated clustering process. Clusters are shown as large circles and there are six objects to be clustered. For this example, there is a minimum of three clusters and a maximum of four clusters. In the first step of the process, for the minimum cluster radius, the algorithm attempted to cluster using the minimum number of clusters. The algorithm is not successful clustering with three clusters and the given cluster radius, as can be observed by the two points that are not covered. Having failed to cluster with three clusters, the algorithm increases the number of clusters to four, with the same cluster radius and tries to cluster again. Again, the points are not clustered completely. Four is the preset maximum number of clusters for this example, so the algorithm again uses three clusters and increases the cluster radius by a preset amount. With this larger cluster radius, the algorithm is successful in covering the data.

2.2 The Performance Matrix

It is useful to provide a measure of the clustering algorithm's success for cases in which the true emitter tracks are known. This is done by defining the performance matrix $\boldsymbol{P}$ whose matrix element $\boldsymbol{P}_{ij}$ is the fraction of points of track i falling into cluster j. If the algorithm succeeds completely in finding the emitter tracks, there will be one and only one track per cluster. In this case, the performance matrix will reduce to the identity matrix. When clustering is less successful, the performance matrix will have off-diagonal entries.

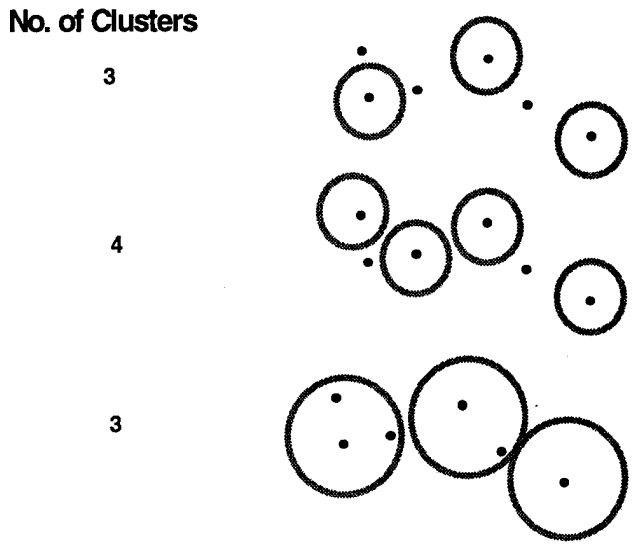

Fig. 5 — An application of the clustering algorithm to the case of six data points to be clustered, for a minimum of three clusters and a maximum of four clusters

2.3 The Speed Discrimination Algorithm

The second component of the ECSDTF algorithm is speed discrimination. This algorithm requires the construction of the speed discrimination matrix, whose elements are defined below. For the k^{th} cluster, let v_{ij}^{k} be the speed that an emitter starting at point i would require to move to point j within the observed time interval. Let v_{min} be the minimum speed, and v_{max} be the maximum speed for this class of emitter carrier. The element in the i^{th} row and the j^{th} column of the speed discrimination matrix for cluster k is denoted as v_{ij}^{k}. Let $v_{ij}^{k} = v_{ij}^{k}$, if $v_{min} \le v_{ij}^{k} \le v_{max}$, otherwise $v_{ij}^{k} = 0$.

In this algorithm, speed discrimination is performed after clustering is conducted. This reduces the speed discrimination matrix to a block diagonal form. The block diagonal form decreases effective dimensionality of the problem, thus diminishing CPU time requirements.

Placing bounds on the speed of an emitter requires judgment. The upper bound, v_{max}, can be obtained from engineering specifications. The lower bound, v_{min}, is 0 knots for a ship since it can remain at rest on the sea. If an aircraft radar is detected, it is probably in the air and not stationary, so one is tempted to take v_{min} as a minimum air speed. The lower bound speed can be less than the minimum air speed as the following examples illustrate: the aircraft is at rest on the ground with its radar on; it is flying in a tight circle or a helix, and seems to be traveling at less than air speed due to the finite sampling rate of the inorganic sensor system; or it is hovering, e.g., a Harrier jump jet. Fortunately, all those examples are improbable.

Speed discrimination reduces the size of matrix elements by effectively eliminating points from clusters. If an emitter at point A would have to move at a speed greater than v_{max} to reach all other points in the cluster, then point A is not connected to the other points in the cluster. When counting the number of track points appearing in each cluster, disconnected single points are eliminated. For points associated with off-diagonal entries in the performance matrix, this is desirable since they represent errors. The

occurrence of a point from track i in cluster j $(i \neq j)$ represents an error, so reduction of these matrix elements is desirable.

Speed discrimination can also reduce diagonal matrix elements. If a point from track i appears in cluster j before speed discrimination, it may be eliminated upon taking speed into consideration. As an example, let q_{ik} be the k^{th} point of track i. For this example, q_{ik} can be connected to q_{ik-1} and q_{ik+1} without traveling at a speed greater than v_{max}, but q_{ik} cannot be connected to any other points of track i without traveling at speeds greater than v_{max}. As such, if q_{ik} appears in a different cluster than q_{ik-1}, and q_{ik+1}, then q_{ik} is a single disconnected point and is eliminated by the speed discrimination algorithm. So speed discrimination can be helpful in forcing the incidence of errors to be less than a certain threshold, but it can also introduce an error in the incidence of track detection (diagonal elements).

2.4 Track Formation

The final step in the ECSDTF algorithm is track formation, i.e., the construction of the line segments making up the potential tracks. The line segments are formed by connecting each pair of points within a cluster for which the associated matrix element of the speed discrimination matrix is non-zero.

If the sampling rate is high, then there may be many potential paths. In such instances, it is frequently useful to only plot the path formed by connecting each cluster point to its nearest neighbor in time that resides in the same cluster. In forming the track, no point is permitted to have more than one input arrow or output arrow. If more than one path can be formed under these constraints, select the path with the maximum number of points. If more than one path still exists for a collection of points that satisfy the above conditions, select the first one formed. It may not be possible to connect all the points in one cluster by a single path because of restrictions imposed by speed discrimination. Regardless of whether the process of connecting nearest neighbors in time within a cluster results in a single path or many paths, the resulting curves will be called *time-nearest-neighbor tracks.*

2.5 Recursive Extensions

The tracking algorithm described up to this point is a batch algorithm, in the sense that before using it, a certain amount of data is allowed to accumulate. By adding a simple post-processing routine, the algorithm can be rendered recursive. The first post-processing step consists of determining the distance between each new arrival and the nearest established cluster center. If the distance is less than the cluster radius then the point is assigned to the associated cluster; otherwise, it becomes the center of a new cluster. For the second processing step, each new data point that falls into a previously established cluster results in a new row and column being added to that cluster's speed discrimination submatrix. If the point is the center of a new cluster, then a new block is added to the speed discrimination matrix as previously described. Finally in the last post-processing step, the tracks are redrawn. The three post-processing steps will allow the algorithm to update the existing tracks, without redoing calculations, rendering the algorithm recursive.

3. APPLICATION OF THE TRACKING ALGORITHM TO BOTH SIMULATED AND EXPERIMENTAL DATA

In this section, ECSDTF is applied to synthetically generated data and measured data for a ship and an aircraft. In all cases, the data sets have the same attributes. First, the simulated ship data is examined, since the ground truth is known and the performance matrix $\boldsymbol{P}$ can be calculated.

Four ships with tracks that have longitude as a linear function of latitude are selected so that two tracks intersect and the remaining two share no common point. A uniform random variable is added to each track latitude and longitude value to produce small departures from linearity.

Figure 6 is a latitude-longitude representation of the tracks of the four simulated ships. Tracks are labeled as T1, T2, T3, and T4. Each track has 20 sample points labeled with circles. Longitude ranges from 24° to 31°and latitude from 58.5° to 63.5°. Tracks are essentially linear, with small fluctuations generated using uniform random variables.

The RF and PRI are also simulated. For each track, the mean and standard deviations of the RF and PRI are estimated from ship data. Let the notation *mean(X)* and *std(X)* be the mean and standard deviation of X, where $X=RF_i$ or PRI_i. The i-subscript specifies values of the random variable for the i^{th} track. For the i^{th} track, RF and PRI are given by

$$RF_i = mean(RF_i) + std(RF_i) \times N(0,1)$$

$$PRI_i = mean(PRI_i) + std(PRI_i) \times N(0,1)$$

for i = 1,2,3,4, where *N(0,1)* is a zero mean, unit variance Gaussian random variable.

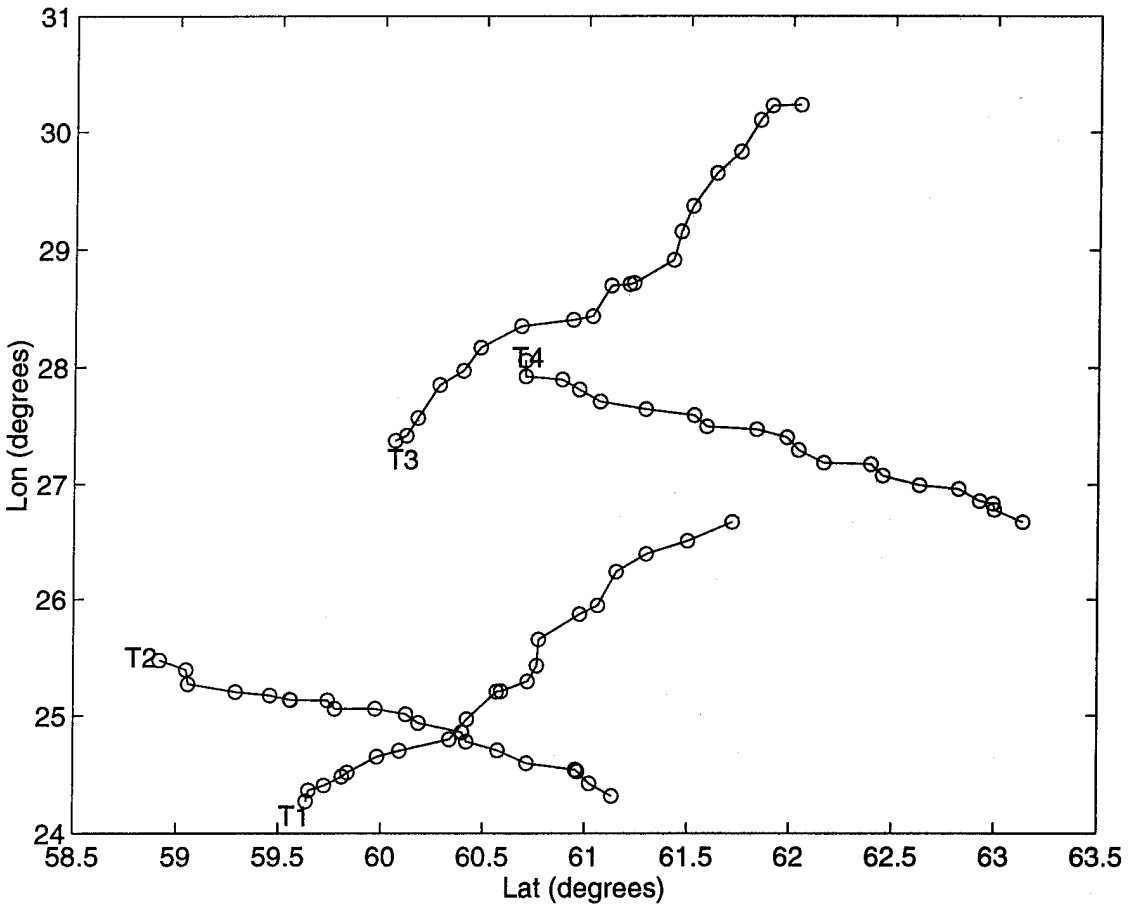

Fig. 6 — Four simulated ships with two tracks crossing and two nonintersecting. The tracks are labeled as T1, T2, T3, and T4.

Figure 7 represents the results of clustering on simulated RF and PRI. The points of each cluster have a unique shape. The clustered points are represented by +'s, *'s, circles, and x's. There are four distinct clusters with PRI and RF spanning 80 μs and 40 MHz, respectively. The cluster of +'s has 39 points; the cluster of circles, 19 points; the cluster of *'s, 2 points; and the cluster of x's, 20 points.

Success in clustering is dependent on the localization of points in the RF-PRI plane. If the clusters are nonintersecting and sufficiently well separated, then each track will be isolated in a single cluster and the algorithm will be completely successful, in which case the performance matrix (Section 2) is an identity matrix.

The clustering algorithm will fail to distinguish individual tracks, if values of RF and PRI are nearly the same. A trivial example of this is, if each track has the same common constant value of RF and PRI for each sample point, then there could only be one cluster and no hope of distinguishing tracks based on clustering. If clustering fails to separate tracks, speed discrimination may produce distinct tracks, if they are sufficiently well separated in latitude and longitude.

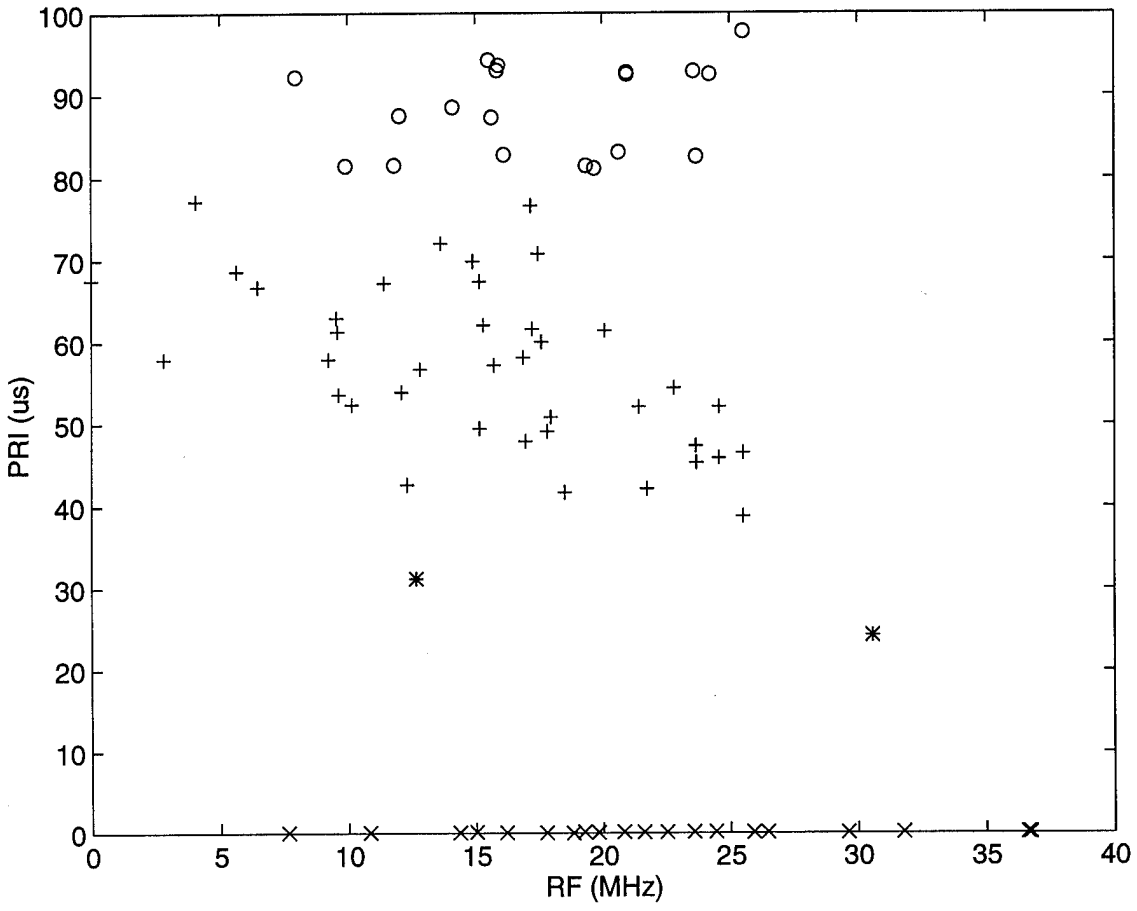

Fig. 7 — Results of clustering on simulated RF and PRI for the simulated ship tracks in Fig. 6. The points of each cluster have unique shape. The cluster points are represented by +'s, *'s, circles, and x's.

Figure 8 is the time-nearest-neighbor-track, one of many potential paths found in cluster 4. Points with a higher numerical label come later than those with a smaller label, e.g., point 1 comes before point 2, point 2 before point 3, etc. This potential track contains all 20 points of track 4, and in this sense, the algorithm was completely successful in track reproduction.

Figure 9 displays cluster 2, which consists of 18 points from track 2, the missing points being 7 and 20. The points have been connected by the time-nearest-neighbor track. The algorithm is successful in reproducing 90% of track 2.

Figure 10 is a representation of cluster 3, which consists of two points from track 1. This is an example of how the algorithm can make an error.

In Fig. 11, cluster 1 is shown to consist of time-nearest-neighbor tracks from tracks 1, 2, and 3. There are 17 points from track 1, 2 points from track 2, and 18 points from track 3.

In this figure, the clustering algorithm was not successful in separating the points of the three tracks into separate clusters. The speed discrimination plus track formation algorithms are successful in forming three separate tracks (subclusters) as is observed. In this sense, speed discrimination plus track formation can be regarded as a simple graph-theoretic clustering algorithm.

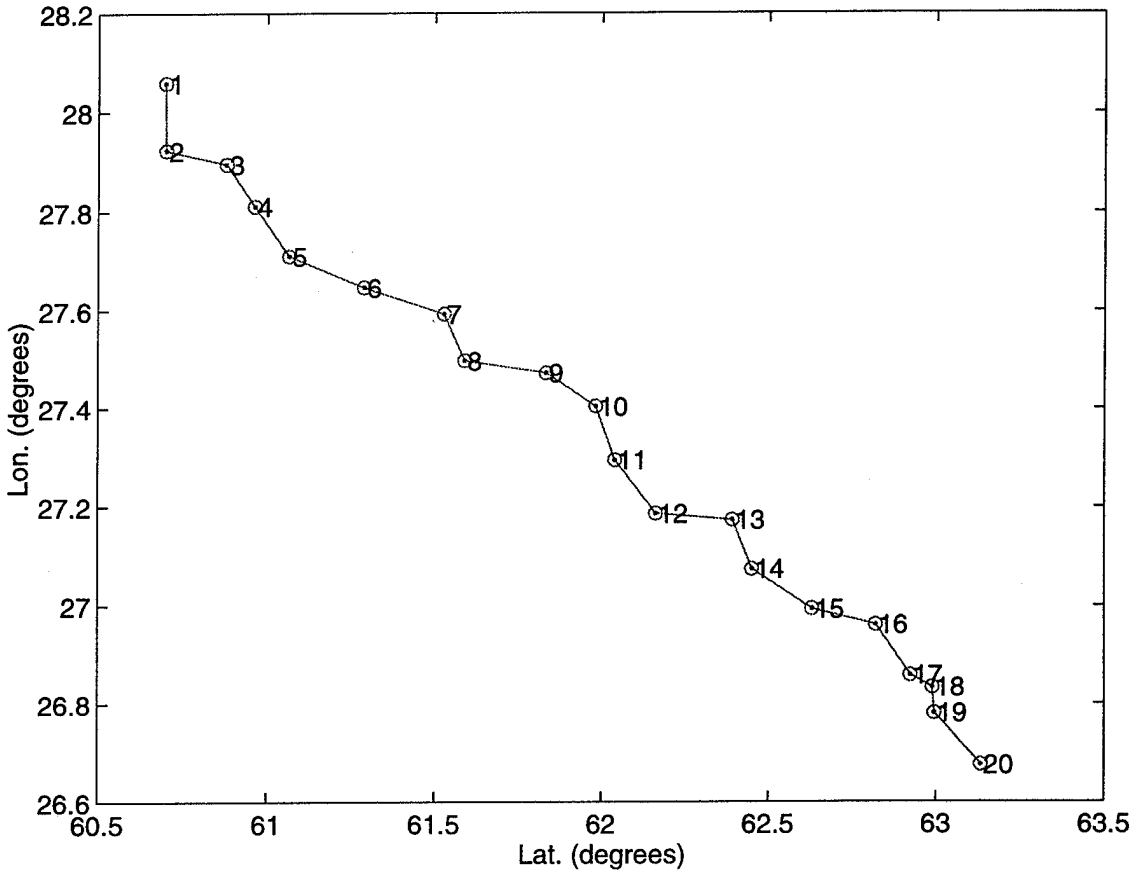

Fig. 8 — The time-nearest-neighbor track for cluster 4 for the simulated ship case. For this example, cluster 4 contains all the points of track 4.

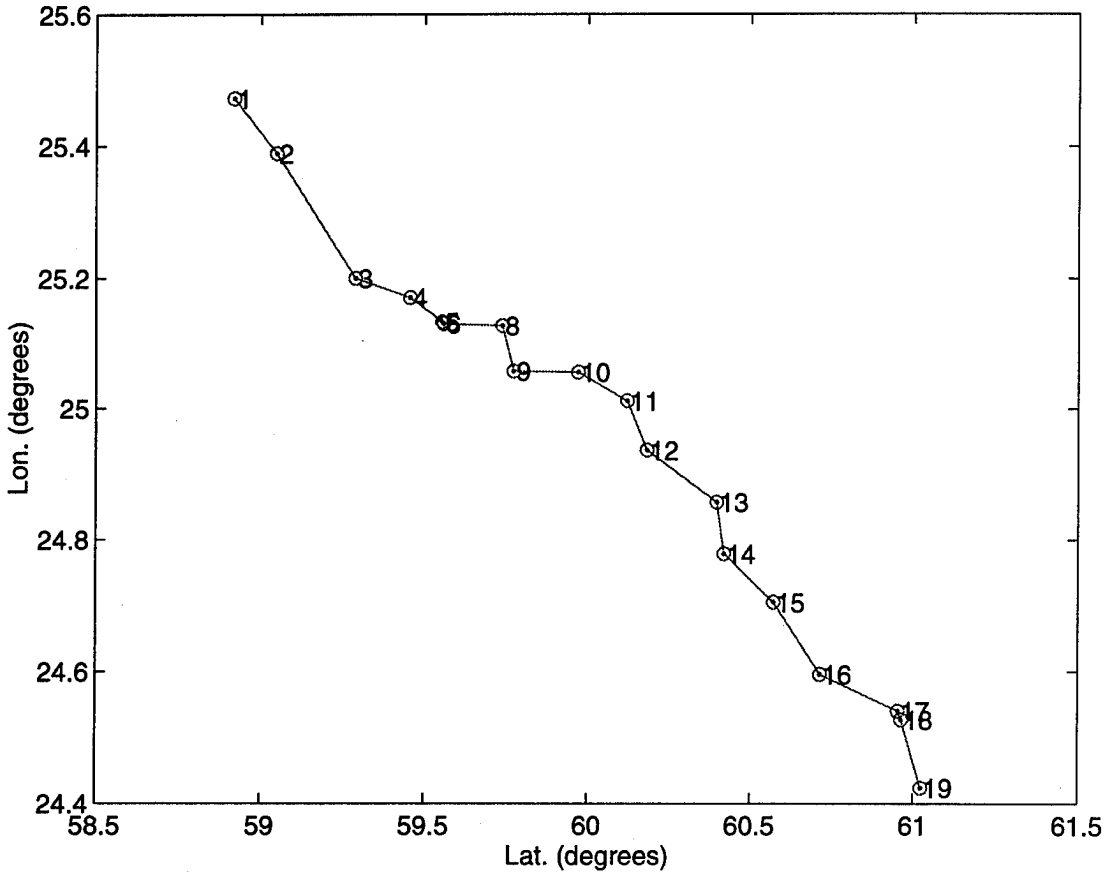

Fig. 9 — The time-nearest-neighbor track for cluster 3 for the simulated ship case. For this example, cluster 3 contains 95% of the points of track 2.

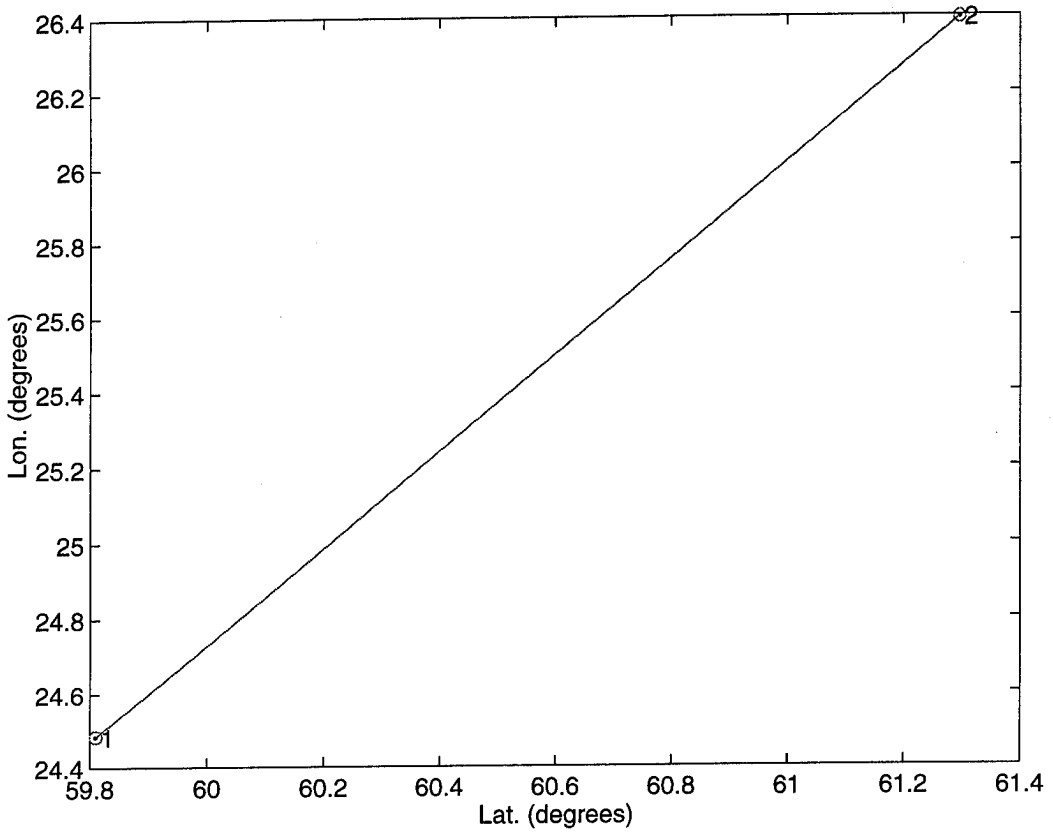

Fig. 10 — A potential track in cluster 2 with two points of track 1 for the simulated ship case. This is an example of how the clustering algorithm can give rise to anomalous paths.

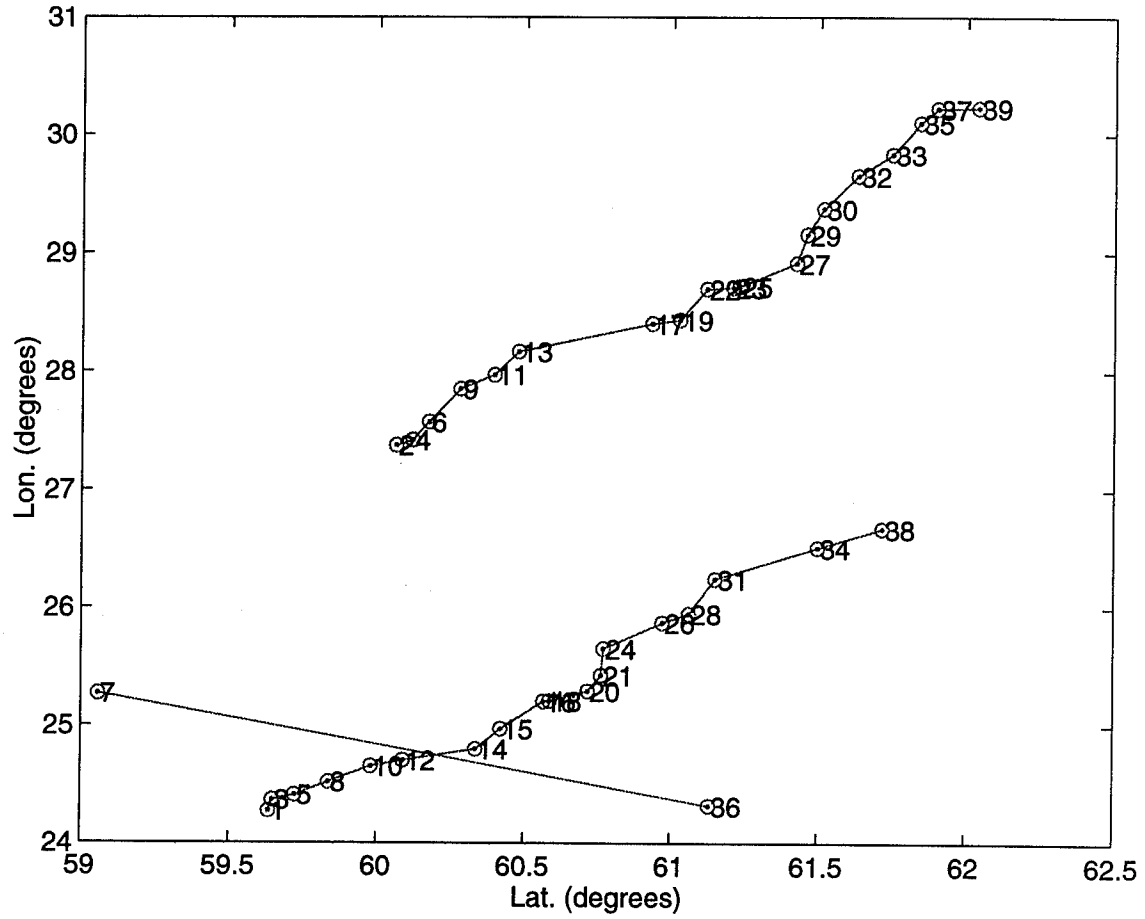

Fig. 11 — The time-nearest-neighbor tracks found in cluster 1 for the simulated ship case. There are 17 points from track 1, 2 points from track 2, and 18 points from track 3.

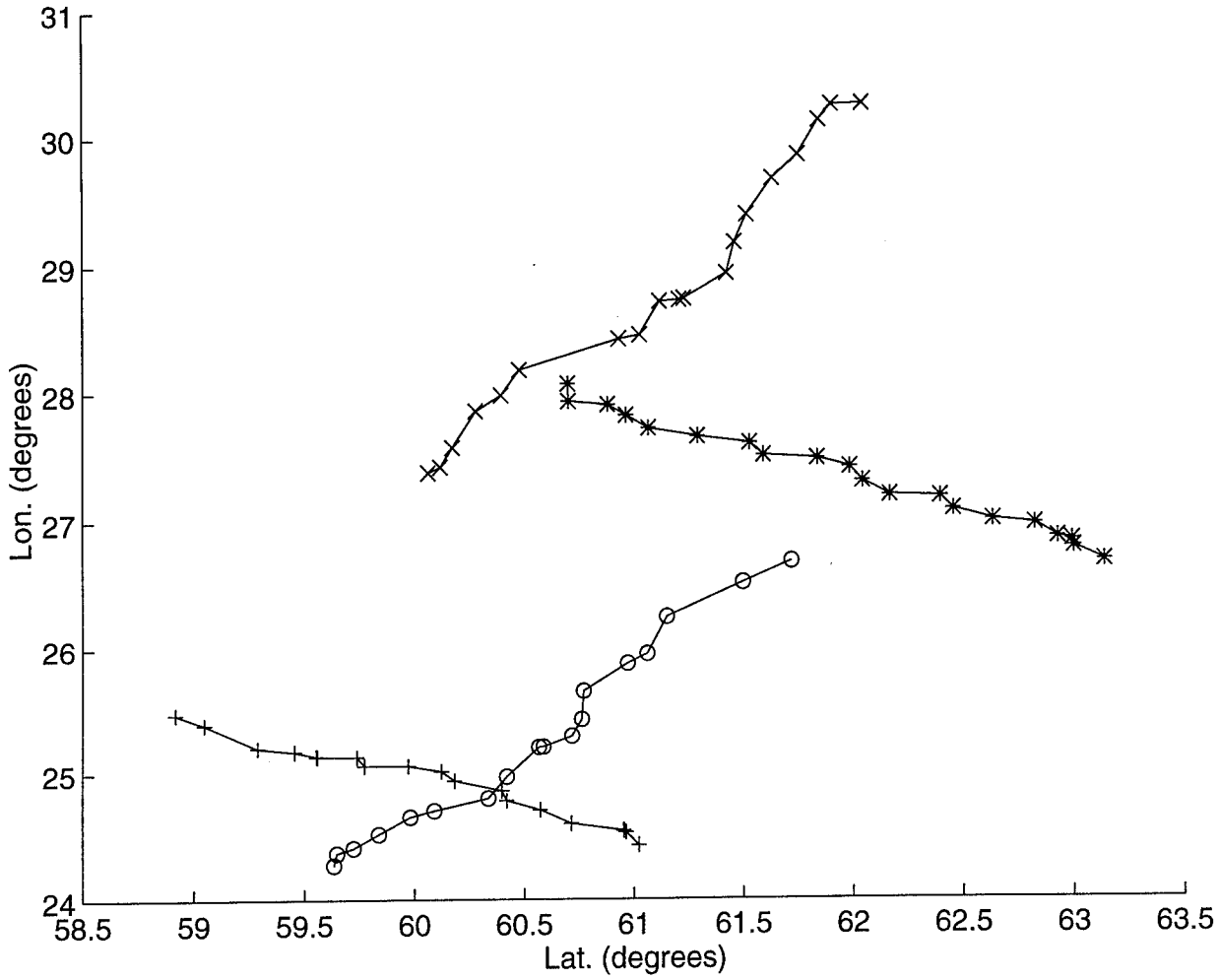

Fig. 12 — The four longest time-nearest-neighbor tracks extracted by the ECSDTF algorithm for the case of four simulated ships

In Fig. 12, the four longest time-nearest-neighbor tracks from Figs. 8 through 11 are displayed. Two, two-point tracks found in Figs. 10 and 11 have been deleted based on the *a priori* assumptions that there were four targets and that the longest tracks have the greatest likelihood of being associated with the emitters. Comparison between Figs. 6 and 12 shows that the ECSDTF algorithm has reproduced most of the features of the original tracks.

The speed discrimination and track formation algorithms partially make up for the errors introduced by Euclidean clustering, but as can be observed from Figs. 8 through 12, there is significant room for improvement. A clustering algorithm that confines all the points of each track to one and only one cluster is desirable, especially when dealing with data sets with a small number of points as is frequently encountered in real-world situations. If certain points from a track become separated from the remainder of the track because of a failure of the clustering algorithm, it can be improperly concluded that there are more targets present than exist in reality. Also, if points are improperly clustered, and subsequently fail to be connected to other points in the cluster due to speed discrimination, they can be lost.

It is essential that a new clustering algorithm be developed. This algorithm must be optimal in its ability to establish data point cluster assignments, especially in the case when a point falls on the

boundary between clusters. An essential feature that the new clustering algorithm must possess is the capability of indicating the grade of membership of each point in each cluster. This will provide a confidence measure for the clustering operation. A fuzzy clustering algorithm with these features is currently under examination.

Tables 1 and 2 give the fraction of points from each track appearing in each cluster given 100 iterations. Table 1 represents results for clustering without speed discrimination, whereas Table 2 is a clustering plus speed discrimination result. When speed discrimination is included, the entries are always less than or equal to those where clustering is used.

Tables 1 and 2 provide a measure of partial success of the algorithm. If the clustering is totally successful, then the numerical 4×4 submatrix would be diagonal with unity in each diagonal entry. As can be readily observed from the matrices, the algorithm is not a complete success. It only reproduced tracks 1 and 4 at a 91% level, independent of whether speed discrimination was included. The algorithm reproduced track 2 at the 53% level. Track 3 is spread out over three clusters, with 60% falling into cluster 1, 28% into cluster 2, and about 11% into cluster 3. For all four tracks, the results are essentially independent of speed discrimination.

The fractional quantities in Tables 1 and 2 can also be interpretted as the degree of compatibility of each track with a specific cluster. This is a concept from fuzzy set theory [9], which is useful because of the intuitive interpretation it provides.

Table 1 — Percentage of Track Points in a Cluster Before Speed Discrimination

	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Track 1	.917	.083	.000	.000
Track 2	.053	.536	.412	.000
Track 3	.603	.280	.118	.000
Track 4	.000	.003	.083	.915

Table 2 — Percentage of Track Points in a Cluster After Speed Discrimination

	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Track 1	.917	.074	.000	.000
Track 2	.053	.533	.411	.000
Track 3	.603	.274	.107	.000
Track 4	.000	.003	.083	.913

Figure 13 is the result of clustering data measured by an inorganic sensor system. The measured data typically has IDs corresponding to several types of emitters. Prior to clustering, a subset of the data is selected by fixing the values of emitter ID and PW. Although the ID and PW of the emitter are fixed, the data can still represent several different ships. The points of each cluster are marked by differently shaped symbols. There are eight clusters in this example. PRI and RF span 120 μs and 70 MHz, respectively.

Unlike the simulation case, the number of ship tracks contained in the data is not known, so there is some question about how to set the minimum and maximum number of clusters. For this case, the minimum number of clusters is set at three and the maximum number at 10, which results in EC converging to eight clusters. The final number of clusters EC determines to be present is fairly insensitive to the initial selection of N_{min} , N_{max} , and Δr. If the clustering algorithm were perfect, eight clusters would correspond to eight ships. In reality, once speed discrimination and track formation are carried out, there can be more or fewer than eight ships present.

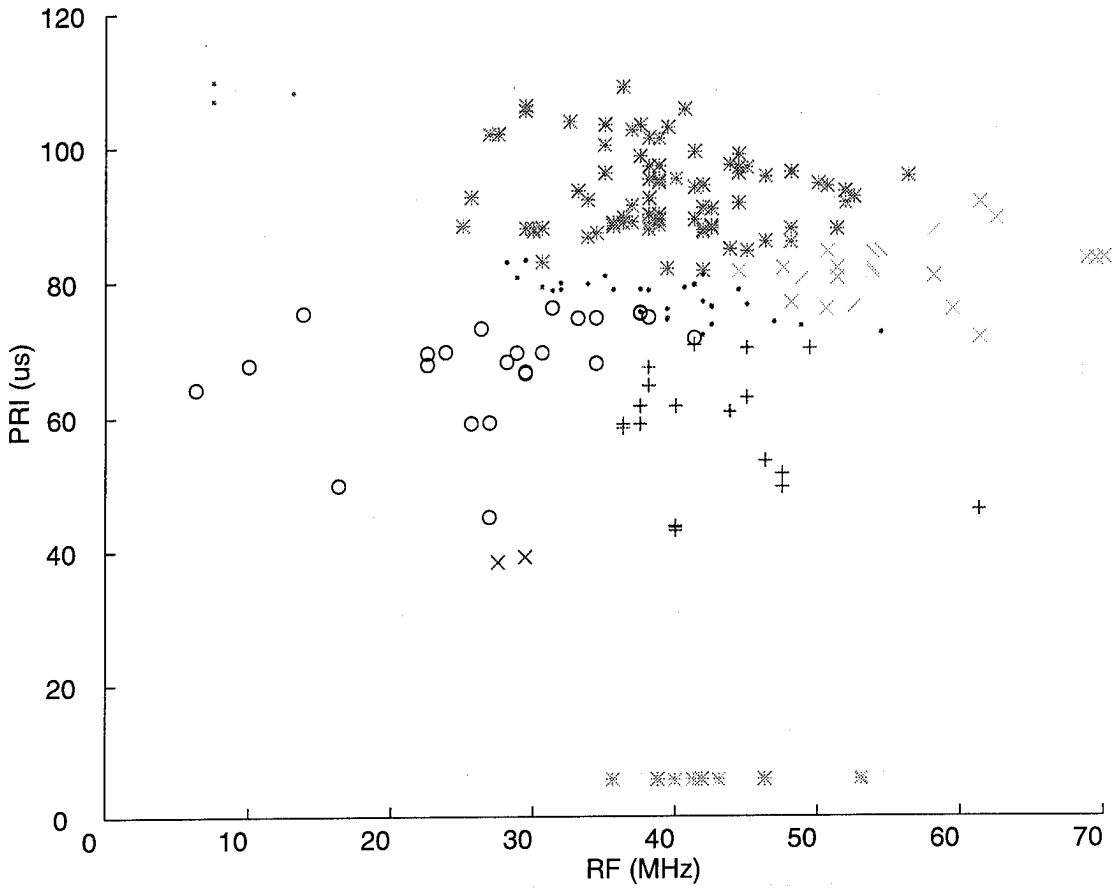

Fig.13 — Clustered results for ship data, measured with an inorganic sensor system. The points of each cluster are marked by differently shaped symbols.

Figure 14 represents the potential ship tracks found in the first cluster after speed discrimination. As in Figs. 8 through 11, time increases with increasing numerical label. The arrows also represent the direction of time flow. Two point paths connect points 20 and 70, points 20 and 79, 20 and 69, etc. Four-point and, subsequently, three-point subpaths are formed by connecting points 34, 39, 69, and 79. As in the case shown in this figure, no cluster has a track with more than four points.

The number of potential tracks and the number of points making up a track is related to the maximum permitted number of clusters. This number is established in the clustering algorithm's input, and should be related to the number of emitters present which is generally not known *a priori*. It may be possible to get potential tracks with more emitters on them by reducing the maximum number of clusters. Experimentation in this case does not reduce significantly the number of emitter points on potential tracks when the maximum number of clusters is decreased from 10 to 4.

The number of potential tracks within a cluster is also influenced by sampling rate. It is possible to simulate a very high sampling rate, e.g., 20 points per track. In the example considered in Fig. 6, this results in many potential tracks. The time-nearest-neighbor track is the only one displayed. Sampling rates are typically much smaller for real data, as is the number of potential tracks.

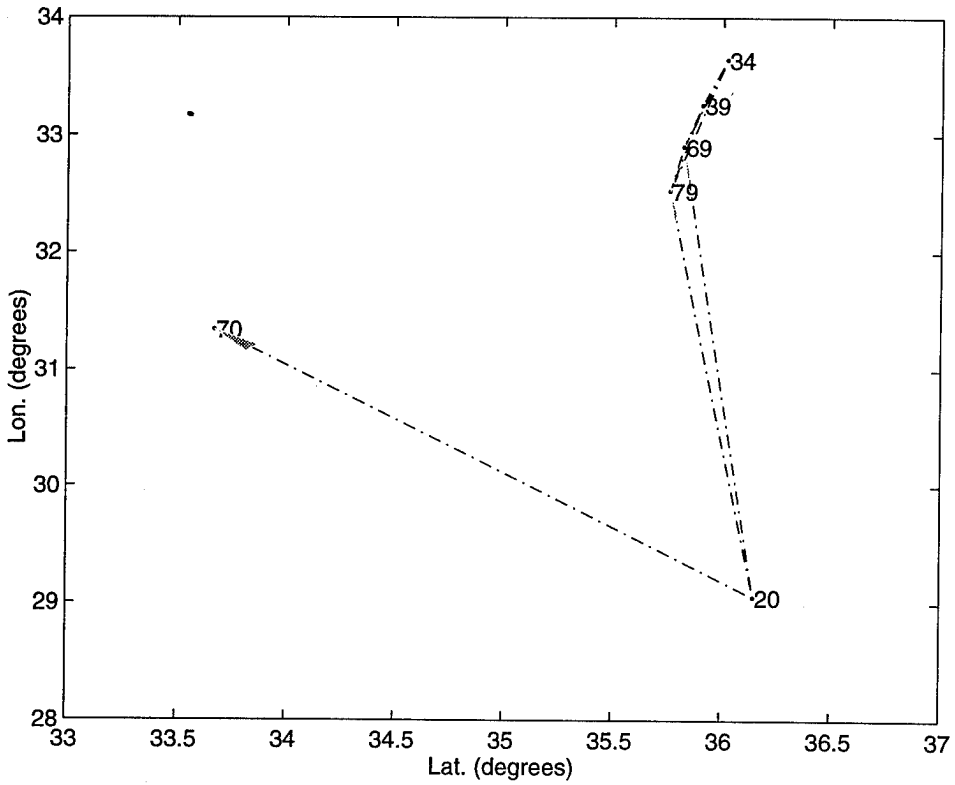

Fig. 14 — Potential ship tracks found in cluster 1 after speed discrimination

Figure 15 shows all potential tracks appearing in cluster 2 for the same ship as in Fig. 13. The number of potential tracks is greater in this plot and four point tracks are observed.

Figure 16 displays all of the time-nearest-neighbor tracks extracted by ECSDTF for the real ship data. Latitude ranges from 10° to 80° and longitude from -100° to 200°. The scale of the latitude-longitude window necessary to display all the tracks results in individual tracks appearing as a small collection of points. Seven markers are used to distinguish points of tracks. Although the same markers are used for different tracks, the separation between similarly marked tracks in latitude and longitude is sufficient that reuse of markers is never confusing.

Figure 17 represents the RF-PRI clustered data points for an aircraft as measured by the inorganic sensor system used in Fig. 13. Emitter ID and PW are fixed. There are six clusters, with each having three to five points. PRI and RF span 40 μs and 70 MHz, respectively.

Figure 18 shows all potential tracks found in the first cluster. Each track has at most three points. Other clusters exhibit the same density of tracks, with at most three points on each track.

The failure of points 1 and 5 in Fig. 18 to form a track segment illustrates the effect of assuming a minimum airspeed. The minimum acceptable airspeed for this class of emitter is found in the literature to be 100 knots. A direct calculation of the speed required to connect points 1 and 5 gives 42 knots, too low to form a track segment.

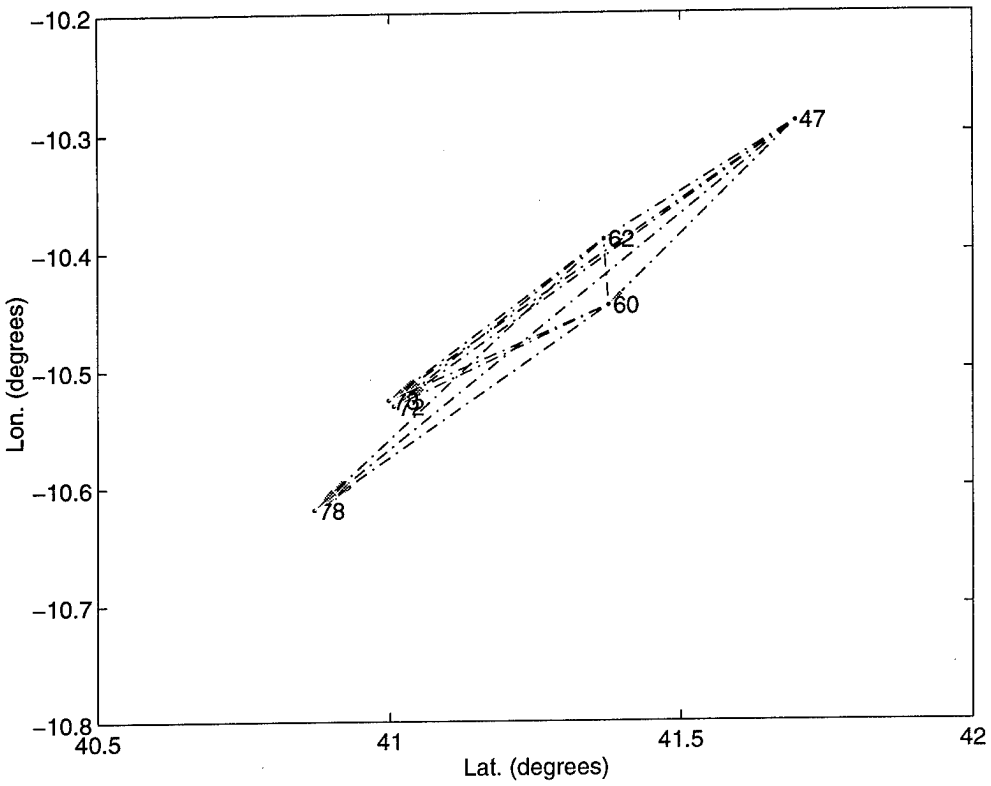

Fig. 15 — Potential ship tracks found in cluster 2 after speed discrimination

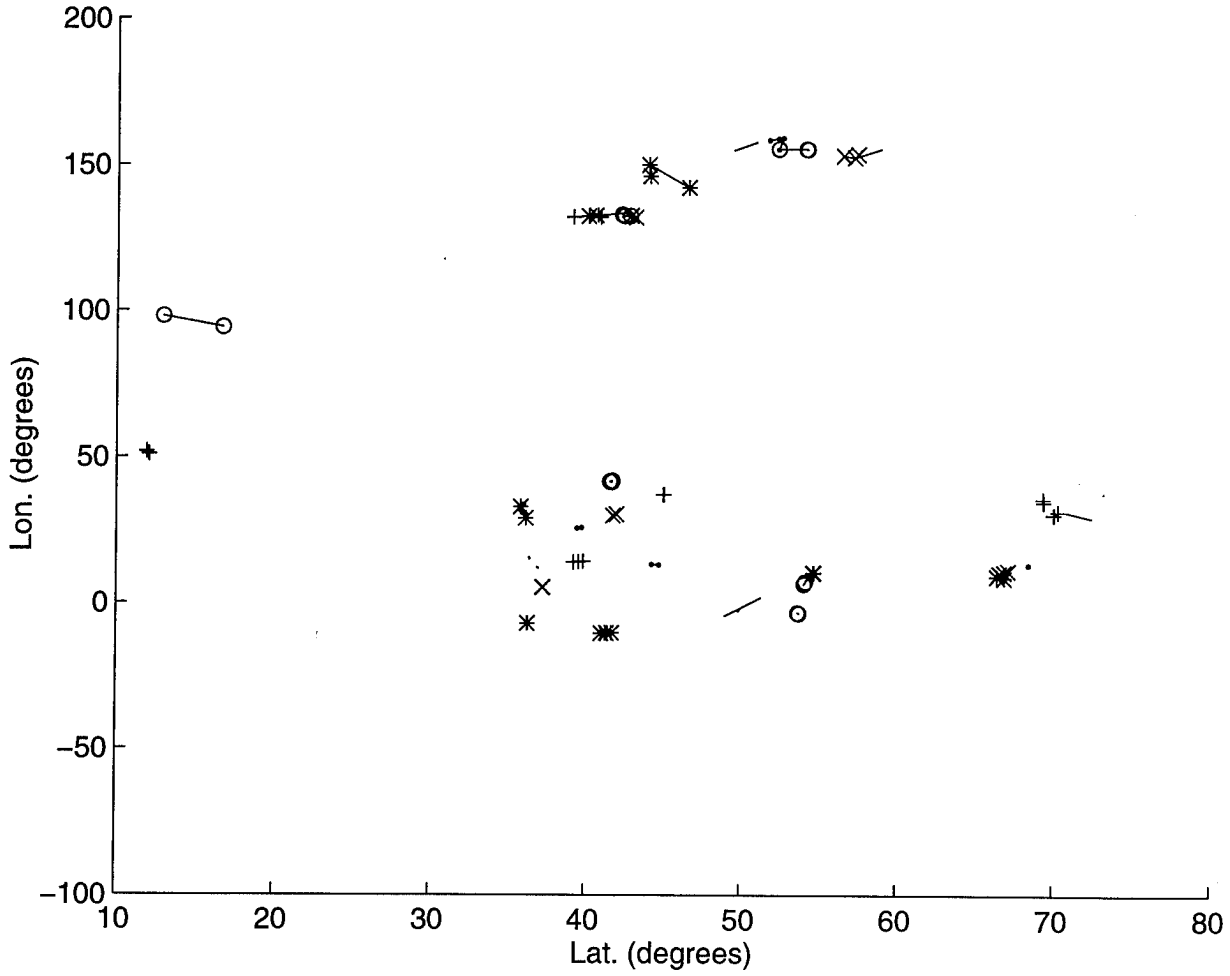

Fig. 16 — The time-nearest-neighbor-tracks found for the measured ship data

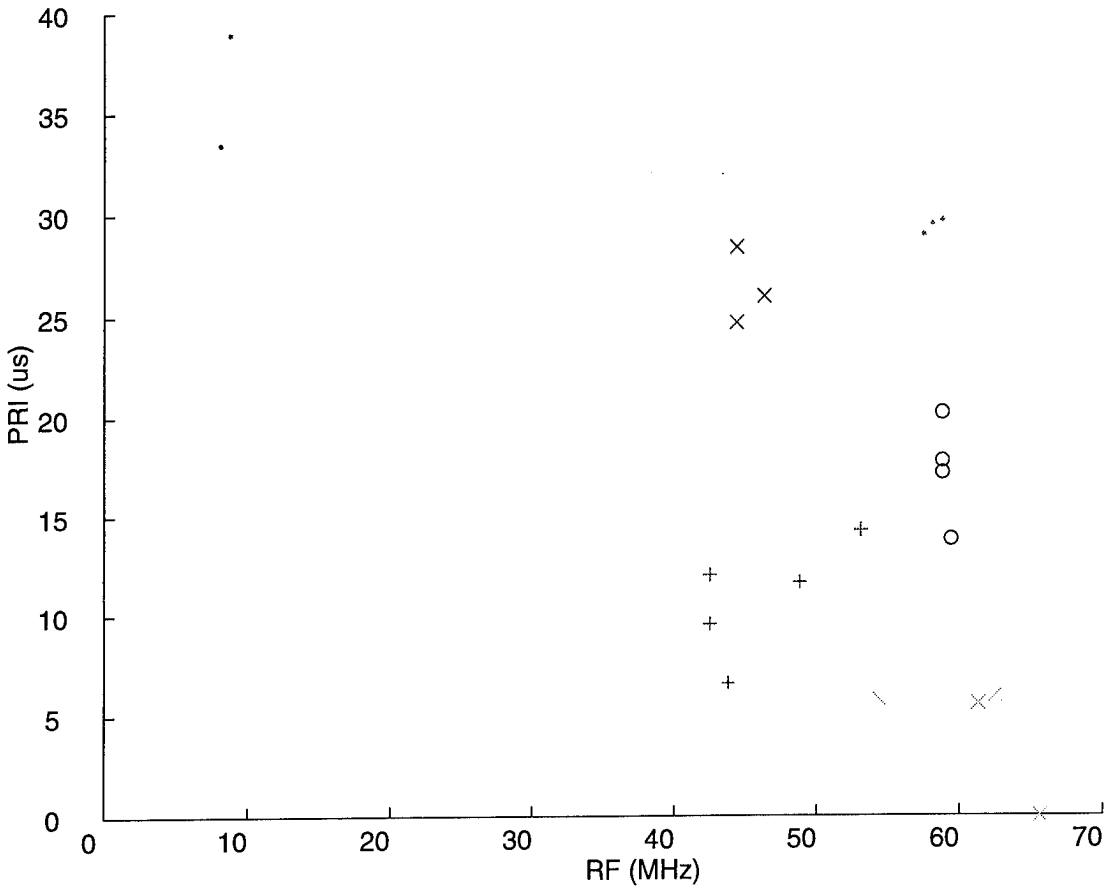

Fig. 17 — Clustered results for aircraft data, measured with an inorganic sensor system. The points of each cluster are marked by differently shaped symbols.

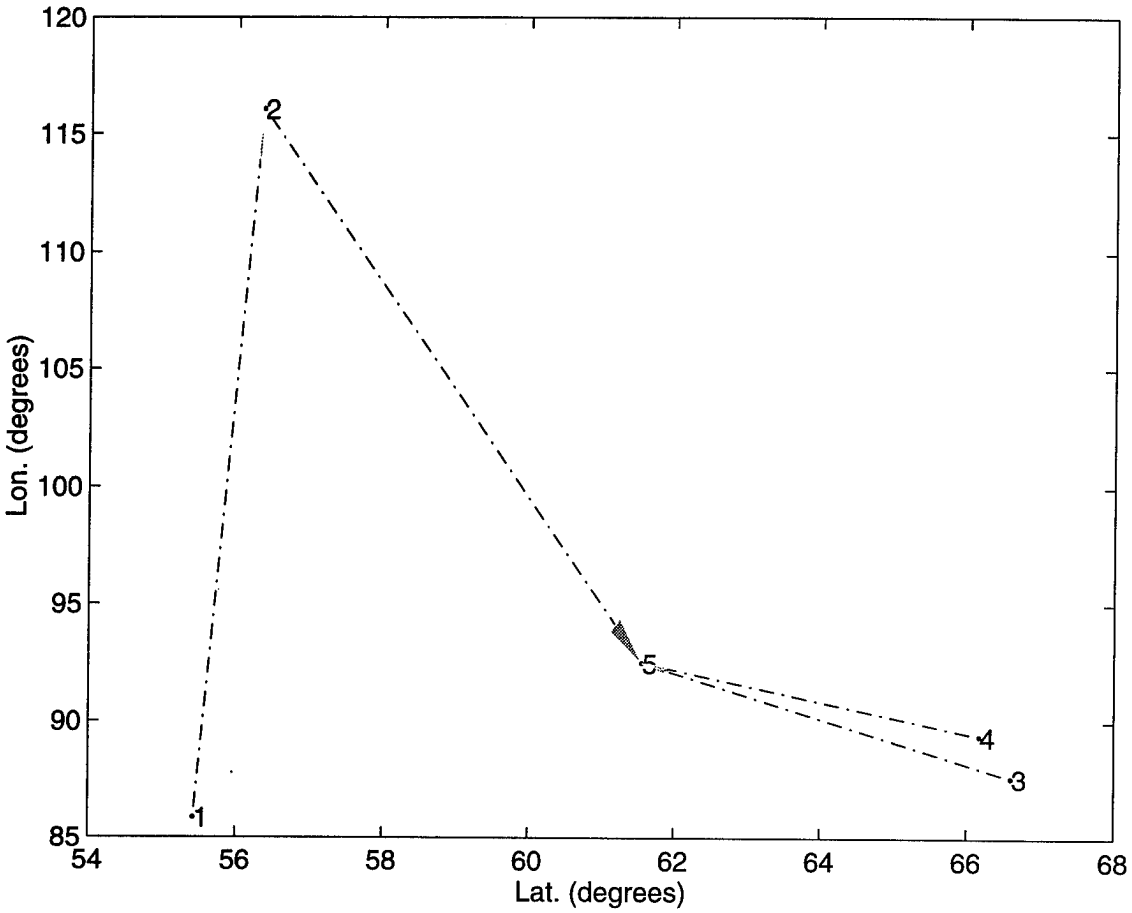

Fig. 18 — All the potential aircraft tracks found in the first cluster

Figure 19 displays all 6 of the time-nearest-neighbor tracks found by ECSDTF for the aircraft data. There are two 2-point tracks and four 3-point tracks in this 30° by 100° latitude-longitude window.

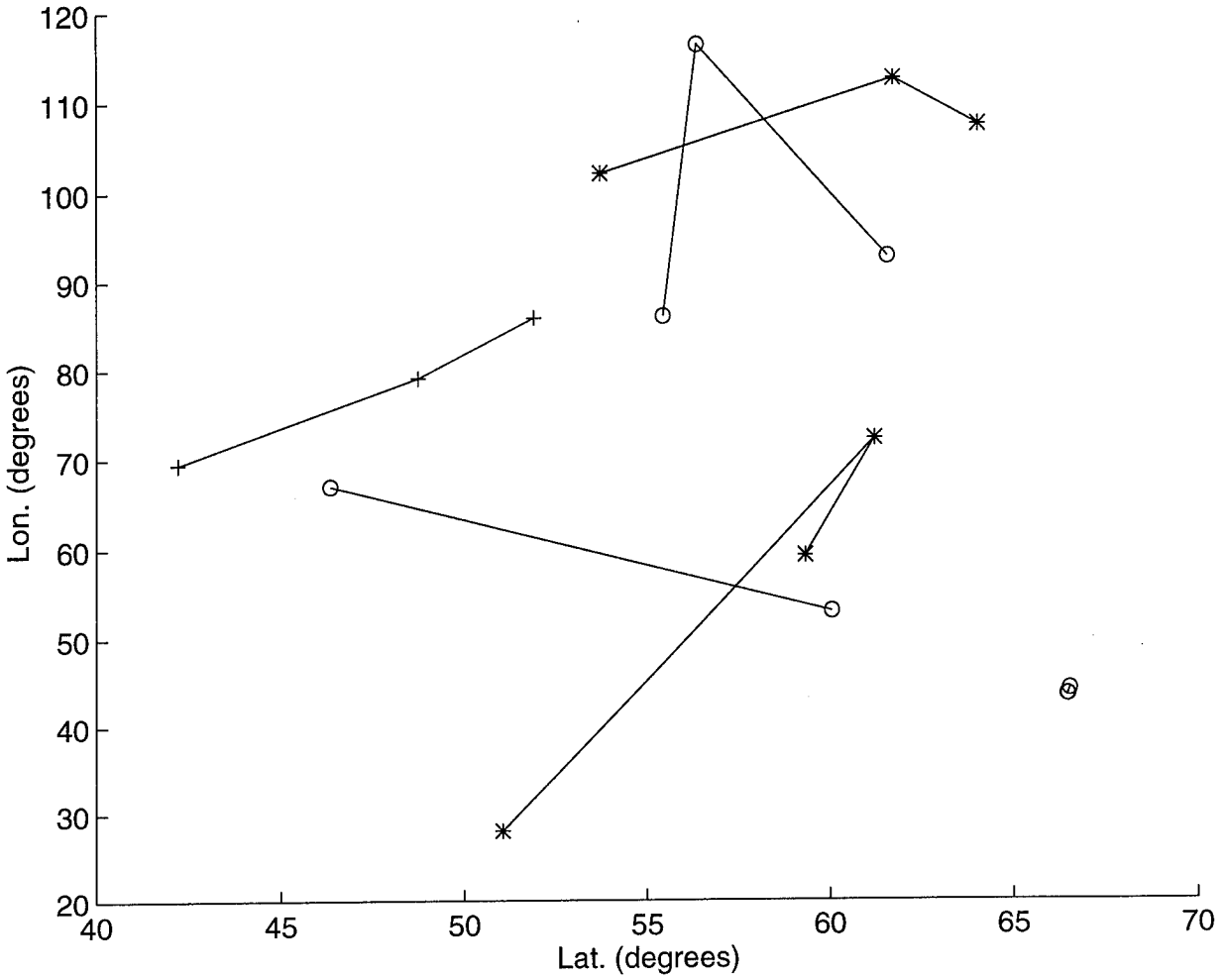

Fig. 19 — The six time-nearest-neighbor tracks extracted by the ECSDTF algorithm for the observed aircraft

4. CONCLUSIONS

Euclidean clustering, speed discrimination, and track formation algorithms are applied to an ESM tracking problem. Euclidean clustering is successful in extracting tracks, if different emitters are well separated in the RF-PRI plane. For complete success, the performance matrix reduces to the identity matrix. For cases where emitter tracks have overlapping regions in the RF-PRI plane, the algorithm will generally be less successful and the performance matrix will either become banded or depart significantly from a diagonal form (an exception is discussed below).

Speed discrimination and track formation can improve track extraction, when the RF-PRI clustering results are poor. Speed discrimination will reduce off-diagonal elements in the performance matrix. If points from the same track fall into more than one cluster, speed discrimination can also reduce diagonal matrix elements, which is observed to be an unfortunate but small effect. If clustering alone is incapable of extracting tracks, speed discrimination can be useful, when the tracks are sufficiently well separated in

latitude and longitude. In the most extreme example of this kind, all tracks will fall into the same RF-PRI cluster (the performance matrix has unity along one column and zeros elsewhere), but by forming time-nearest-neighbor paths, the tracks can be extracted. In this sense, the final track formation process can be regarded as a graph-theoretic clustering algorithm.

For high sampling rate examples (as in simulations), the number of potential paths is large, suggesting the use of the time-nearest-neighbor path. Tracks with more than four points are rarely observed in experimental data, so all potential tracks are plotted.

The low observational density per unit time provided by the inorganic sensor system suggests a wait time measured in seconds or minutes before application of the tracking algorithm. Once tracks are formed, the recursive extension of the algorithm will allow tracks to be updated in real time as new data arrives without redoing calculations.

For systems that have a much higher obervational density per unit time, such as ESM systems, the wait time for accumulation of sufficient data may be measured in seconds. Then the ECSDTF algorithm can be implemented in real- or near real-time, depending on processor speed. The recursive extension remains valuable since it allows tracks to be updated without redoing calculations, thus speeding up the final stage of analysis.

5. FUTURE EXTENSIONS OF THE TRACKING ALGORITHM

It is possible to improve the tracking algorithm both at the level of clustering and discrimination. Also, simulations could be made that are significantly more sophisticated.

It is essential that a new clustering algorithm be developed. This algorithm must be optimal in its ability to establish data point cluster assignments, especially in the case when a point falls on the boundary between clusters. An essential feature, the new clustering algorithm must possess is the capability of indicating the grade of membership of each point in each cluster, this will provide a confidence measure for the clustering operation. A grade of membership would also be useful for track defragmentation, i.e., a post-processing step where tracks whose points fall into more than one cluster are put back together. A fuzzy clustering algorithm with these features is currently under development.

Another useful feature the new clustering algorithm should possess is easy incorporation of *a priori* information (i.e., information produced by a processing step prior to the main clustering operation). This is also an aspect of the fuzzy clustering algorithm currently under development.

Additional discrimination rules might be pursued, such as

- Kinematics other than translational velocity, such as translational acceleration, angular speed, or angular acceleration;
- Geographical discrimination, e.g., exclusion of line segments connecting two ship data points on the opposite sides of a land mass;
- Energetic discrimination, e.g., a ship's captain selecting the most energetically efficient route unless there was something to be gained by taking an energetically less efficient route.

REFERENCES

1. S. S. Blackman, *Multiple-Target Tracking with Radar Applications* (Artech House, Inc., Boston, MA, 1986).

2. Y. Bar-Shalom and X. Li, *Estimation and Tracking: Principles, Techniques, and Software* (Artech House, Inc., Boston, MA, 1993).

3. Y. Bar-Shalom and T. E. Fortman, *Tracking and Data Association* (Academic Press, Inc., San Diego, CA, 1988).

4. Y. Bar-Shalom, *Multitarget-Multisensor Tracking: Applications and Advances,* Vol. I (Artech House, Inc., Boston, MA, 1989).

5. Y. Bar-Shalom, *Multitarget-Multisensor Tracking: Applications and Advances,* Vol. II (Artech House, Inc., Boston, MA, 1989).

6. D. L. Hall, *Mathematical Techniques in Multisensor Data Fusion* (Artech House, Inc. Boston, MA, 1992).

7. C. Bachman, P. Bey, J. Broughton, V. Chen, S. Gardner, B. Kamgar-Parsi, B. Kamgar-Parsi, M. Kim, F. Kub, K. Moon, A. Schultz, J. Sciortino, D. Scribner, A. Skinner, W. Tolles, J. Willey, and S. Wolk, "Neural Networks: An NRL Perspective," Naval Research Lab Technical Report, NRL/MR/1003--93-7396.

8. H. Spath, *Cluster Analysis Algorithms for Data Reduction and Classification of Objects* (John Wiley and Sons, New York, 1980).

9. T. Terano, K. Asai, and M. Sugeno, *Fuzzy System Theory and its Applications* (Academic Press, Inc., Harcourt Brace Jovanovich, New York, 1987).